(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 09305440 A

(43) Date of publication of application: 28 . 11 . 97

(51) Int. Cl G06F 11/30
G06F 11/00
G06F 13/00

(21) Application number: 08115984

(22) Date of filing: 10 . 05 . 96

(71) Applicant: MATSUSHITA GRAPHIC COMMUN SYST INC

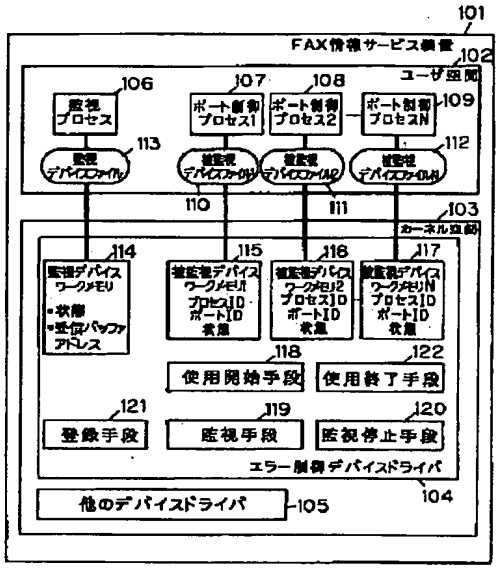
(72) Inventor: SUZUKI TAKU

(54) ERROR PROCESSING METHOD

(57) Abstract:

PROBLEM TO BE SOLVED: To simplify an error process, to mitigate the load of the error process, and to evade a state wherein a device can not be controlled owing to the abnormal end of an operation management process in a processor consisting of plural operating on a multitask monitor and an operating system.

SOLUTION: This method has a monitor process which monitors an error and processes to be monitored; if an error occurs, information to be reported from a monitored process to the monitoring process is previously held before the error occurrence to the monitored process and reported to the monitoring process 106 after the error occurrence to the monitored process. If the abnormal end of the monitoring process 106 is detected during the error monitoring by the monitoring process 108, the system is reset and the monitoring process 106 is started again.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平9－305440
(43)公開日　平成９年(1997)11月28日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ 11/30			Ｇ０６Ｆ 11/30	Ｋ	
11/00	３１０		11/00	３１０Ｃ	
13/00	３５１		13/00	３５１Ｍ	

審査請求　未請求　請求項の数10　ＯＬ　（全 12 頁）

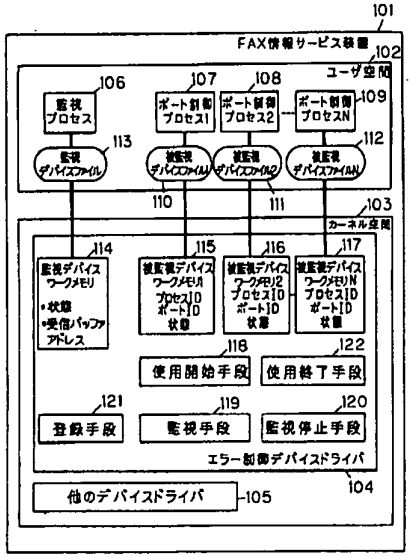
(21)出願番号　特願平8－115984

(22)出願日　平成８年(1996)５月10日

(71)出願人　000187736
松下電送株式会社
東京都目黒区下目黒２丁目３番８号
(72)発明者　鈴木　卓
東京都目黒区下目黒２丁目３番８号　松下電送株式会社内
(74)代理人　弁理士　掩本　智之　（外１名）

(54)【発明の名称】　エラー処理方法

(57)【要約】
【課題】　本発明は、マルチタスクモニタやオペレーティングシステム上で動作する複数のプロセスからなる処理装置において、エラー処理の簡潔化、エラー処理による負荷の軽減と運用管理プロセスの異常終了によって装置が制御不能になる状況を回避することを目的としている。
【解決手段】　エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記被監視プロセスから前記監視プロセスに通知すべき情報を前記被監視プロセスでのエラー発生前に予め保持し、前記被監視プロセスでのエラー発生後に前記情報を前記監視プロセスに通知する、構成とした。また、監視プロセスによるエラー監視実行中に監視プロセスの異常終了が検出された場合には、システムリセットを実行し再び監視プロセスを起動する構成とした。



【特許請求の範囲】
【請求項１】　エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記被監視プロセスから前記監視プロセスに通知すべきエラー情報を前記被監視プロセスでのエラー発生前に予め保持し、前記被監視プロセスでのエラー発生後に前記エラー情報を前記監視プロセスに通知することを特徴とするエラー処理方法。
【請求項２】　システム起動時に呼出されエラー情報を保持する領域を書込み可能な状態にする使用開始手段を有することを特徴とする請求項１記載のエラー処理方法。
【請求項３】　エラー発生の際に前記被監視プロセスから前記監視プロセスに通知すべきエラー情報を保持する処理を被監視プロセス起動時に実行する登録手段を有することを特徴とする請求項1記載のエラー処理方法。
【請求項４】　監視プロセスにより起動された後命令受付け待ちの状態で待機し、エラー情報認識後にそのエラー情報を読み出して前記監視プロセスに通知する監視手段を有することを特徴とする請求項1記載のエラー処理方法。
【請求項５】　エラー発生による被監視プロセス終了時にエラー情報を監視手段に認識させる使用終了手段を有することを特徴とする請求項４記載のエラー処理方法。
【請求項６】　監視プロセスにより起動され監視手段に対して監視停止命令を通知して監視を終了させる監視停止手段を有することを特徴とする請求項４記載のエラー処理方法。
【請求項７】　監視プロセスが通常のイベント処理を実行するスレッドとエラー監視処理を実行するスレッドとを有して構成されることを特徴とする請求項５記載のエラー処理方法。
【請求項８】　エラー監視プロセスと、複数の被監視プロセスと、を有し、この複数の被監視プロセスの異常終了状態を各々検出し、前記被監視プロセスが異常終了した場合に起動し異常終了した被監視プロセスを特定する情報を前記エラー監視プロセスに通知するエラー処理を実行することを特徴とするエラー処理方法。
【請求項９】　エラー監視を行う監視プロセスと、複数の被監視プロセスと、これらのプロセスに対応して各々設けられプロセス実行開始時にオープン状態となりプロセス実行停止時にクローズ状態となるデバイスファイルと、を有し、前記いずれかのプロセスが異常終了しデバイスファイルがクローズ状態となった場合にその異常プロセスを特定する情報を前記監視プロセスに通知することを特徴とするエラー処理方法。
【請求項１０】　監視プロセスによるエラー監視実行中にエラー制御手段が監視プロセスの異常終了を検出した場合には、システムリセットを実行し再び監視プロセスを起動することを特徴とする請求項８又は請求項９記載のエラー処理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、マルチタスクのモニタやオペレーティングシステム上で動作する、複数の処理単位（タスクまたはプロセス）から構成される装置におけるエラー処理方法に関するものである。
【０００２】
【従来の技術】マルチタスク環境上で動作する複数の処理単位から構成される装置においては、各処理単位（プロセス）は、外部からのイベント受信を待ち、受信後にその処理を行うという動作をくり返すようになっている。この種の装置のエラー処理方法は、被エラー監視プロセスがエラーを検知すると、エラー監視プロセスに対して、プロセス間通信等の手段によりエラー通知が行うようになっている。
【０００３】以下、従来のエラー処理方法を具体的に図９、図１０を用いて説明する。図９はエラー監視プロセスの処理フローを示し、図１０は被エラー監視プロセスの処理フローを示している。
【０００４】図９において、監視プロセスは、イベントの受信待ちの状態で待機し（ステップ９０１）、イベントが受信されると受信イベントが被エラー監視プロセスからのエラー通知メッセージか否かをチェックし（ステップ９０２）、エラー通知であれば適切なエラー処理を実行した後（ステップ９０３）、イベントの受信待ちの状態に復帰する（ステップ９０１）。
【０００５】受信イベントが通常のイベントであれば受信イベントの処理を実行する（ステップ９０４）。
【０００６】通常のイベント処理実行後、一定時間が経過したか否かをチェックし（ステップ９０５）、一定時間が経過していない場合は、ステップ９０１に復帰し、一定時間が経過していた場は、エラー監視となる全ての被監視プロセス（Ｎ個）に対してその状態をチェックする以下の処理を実行する（ステップ９０６以下）。
【０００７】まず、カウンタＩを初期化し（ステップ９０６）、カウンタ値がＮ以下であることを確認した後、カウンタＩをインクリメントし（ステップ９０７、ステップ９０８）、被エラー監視プロセスに状態チェックメッセージを送信し（ステップ９０９）、そのメッセージに対するレスポンス受信を待つ（ステップ９１０）。その後、エラーを報告するメッセージを受信した場合や、レスポンス待ちのタイムアウトが発生した場合には（ステップ９１１）、適切なエラー処理を行う（ステップ９１２）。また、ステップ９０９で正常なレスポンスを受信した場合もステップ９１０からステップ９０７へジャンプする。
【０００８】以上一連の処理をカウンタＩがＮに達するまで繰り返し、全ての被エラー監視プロセスに対する処理が終了した場合には、再びイベント受信待ちの状態に

復帰する（ステップ９０１）。

【０００９】一方、エラー監視の対象である、被エラー監視プロセスの処理は、以下のとおりである。図１０において、被エラー監視プロセスはイベントの受信待ちの状態で待機し（ステップ１００１）、イベントが受信されると、そのイベントがエラー監視プロセスからの状態チェックメッセージか否かをチェックする（ステップ１００２）。

【００１０】発生したイベントが、エラー監視プロセスからの状態チェックメッセージでない場合には、通常のイベント処理を実行し（ステップ１００３）、更に、その通常のイベント処理において障害が発生しているか否かをチェックし（ステップ１００４）、正常であればイベント受信待ちの状態に復帰する（ステップ１００１）。障害が検知された場合には、監視プロセスに対してエラー通知メッセージを送信し（ステップ１００５）、適切なエラー処理を実行した後（ステップ１００６）、イベント受信待ちの状態に復帰する（ステップ１００１）。

【００１１】逆に、発生したイベントが、エラー監視プロセスからの状態チェックメッセージである場合には、被エラー監視プロセスはその状態情報を監視プロセスに送信する処理を行う（ステップ１００７）。

【００１２】この様に、エラー監視プロセスが、定期的にエラー監視の対象となる全てのプロセスに対して状態チェックメッセージを送信し、これを受信した被エラー監視プロセスが状態情報を監視プロセスに送信するような監視形態ととる理由は、例えば、被エラー監視プロセスのプログラムの不具合等により、エラーの検出自体を行なえずプロセスが異常終了してしまい異常終了を監視プロセスに通知することができないという事態を回避するためである。

【００１３】

【発明が解決しようとする課題】しかし、上述の従来技術の構成では、エラーを検出するために監視対象となるプロセスと監視プロセスとがメッセージを交換する必要があるため、本来の処理のほかにプロセス間通信処理部やタイマー処理を組み込まなければならず、ソフトウエアの処理が複雑になる。

【００１４】また、監視プロセスが定期的に全ての監視対象プロセスに対して状態チェックメッセージを送信するため、特に監視対象数が大きい場合は計算資源が無駄に使用されることとなる。

【００１５】また、装置管理機能を持つ監視プロセスが異常終了した場合には、他のプロセスが監視プロセスの異常終了を検知し、エラー処理を行わない限り、装置全体が制御不能になるという問題がある。この問題に対して、監視プロセスの障害検知をメッセージによる通知方式で行う場合には、各監視対象プロセスが逆に監視プロセスに対して状態チェックメッセージを送信することになり、ソフトウエアがますます複雑化し、負荷もさらに大きくなることとなる。

【００１６】本発明は、上述の課題に鑑みて為されたもので、マルチタスクモニタやオペレーティングシステム上で動作する複数のプロセスからなる処理装置において、エラー処理の簡潔化、エラー処理による負荷の軽減と運用管理プロセスの異常終了によって装置が制御不能になる状況を回避すること、を目的としている。

【００１７】

【課題を解決するための手段】上述の課題を解決するため、本発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記被監視プロセスから前記監視プロセスに通知すべき情報を前記被監視プロセスでのエラー発生前に予め保持し、前記被監視プロセスでのエラー発生後に前記情報を前記監視プロセスに通知する構成とした。また、監視プロセスによるエラー監視実行中に監視プロセスの異常終了が検出された場合には、システムリセットを実行し再び監視プロセスを起動する構成とした。

【００１８】この構成により、監視プロセスが定期的に被監視プロセスの状態をチェックする必要がなくなり、エラー処理部のプログラム量を飛躍的に減らすことが可能になり、また、エラー検知にプロセス間通信を使用しないので装置の計算資源の浪費を避けることができる。また、運用管理プロセスの異常終了によって装置が制御不能になる状況を回避することができる。

【００１９】

【発明の実施の形態】請求項１記載の発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、を有し、エラー発生時に前記被監視プロセスから前記監視プロセスに通知すべきエラー情報を前記被監視プロセスでのエラー発生前に予め保持し、前記被監視プロセスでのエラー発生後に前記情報を前記監視プロセスに通知するものであり、エラー検出に必要な情報を被監視プロセスとは異なる場所に保持しておき事後的に監視プロセスに通知することにより被監視プロセスが異常終了した場合でも確実にエラー検出を行ない得る。

【００２０】請求項２記載の発明は、請求項１において、システム起動時に呼出されエラー情報を保持する領域を書込み可能な状態にする使用開始手段を有するものであり、これによりシステム起動時に際しては確実にエラーチェックの準備が整うことになる。

【００２１】請求項３記載の発明は、請求項１において、エラー発生の際に前記被監視プロセスから前記監視プロセスに通知すべきエラー情報を保持する処理を被監視プロセス起動時に実行する登録手段を有するものであり、被監視プロセス起動の際にエラー通知に必要な情報の登録が実行されるため、被監視プロセス起動直後にそのプロセスのエラー監視が可能になる。

【００２２】請求項４記載の発明は、請求項１において、監視プロセスにより起動された後命令受付け待ちの状態で待機し、エラー情報認識後にそのエラー情報を読み出して前記監視プロセスに通知する監視手段を有するものであり、監視手段がエラー検出を開始、続行するため、監視手段を呼出すだけで全ての監視対象プロセスで生じたエラーを検知することが可能となる。

【００２３】請求項５記載の発明は、請求項４において、エラー発生による被監視プロセス終了時にエラー情報を監視手段に認識させる使用終了手段を有するものであり、監視手段を実行させることにより、エラー通知に必要な情報を被監視プロセス終了時に確実に監視プロセスに通知することが可能となる。

【００２４】請求項６記載の発明は、請求項４において、監視プロセスにより起動され監視手段に対して監視停止命令を通知して監視を終了させる監視停止手段を有するものであり、監視手段によりエラー監視を開始し、監視終了手段によりエラー監視を中断でき、監視不要な処理に対してのエラー監視、エラー通知を避けることが可能となる。

【００２５】請求項７記載の発明は、請求項５において、監視プロセスが通常のイベント処理を実行するスレッドとエラー監視処理を実行するスレッドとを有して構成されるものであり、監視プロセス１０６がエラー監視をする処理としない処理を明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を実行する必要がなくなる。

【００２６】請求項８記載の発明は、エラー監視プロセスと、複数の被監視プロセスと、を有し、この複数の被監視プロセスの異常終了状態を各々検出し、前記被監視プロセスが異常終了した場合に起動し異常終了した被監視プロセスを特定する情報を前記エラー監視プロセスに通知するエラー処理を実行するものであり、被監視プロセスが異常終了した場合でも確実にエラー検出、エラー通知を行ない得る。

【００２７】請求項９記載の発明は、エラー監視を行う監視プロセスと、複数の被監視プロセスと、これらのプロセスに対応して各々設けられプロセス実行開始時にオープン状態となりプロセス実行停止時にクローズ状態となるデバイスファイルと、を有し、前記いずれかのプロセスが異常終了しデバイスファイルがクローズ状態となった場合にその異常プロセスを特定する情報を前記監視プロセスに通知するものであり、被監視プロセスの状態監視用のデバイスファイルによりエラー検知を確実に行ない得る。

【００２８】請求項１０記載の発明は、請求項８又は請求項９において、監視プロセスによるエラー監視実行中にエラー制御手段が監視プロセスの異常終了を検出した場合には、システムリセットを実行し再び監視プロセスを起動するものであり、監視プロセスが監視中に異常終了した場合に装置全体が再起動するので、必要となるプロセスを全て再生でき、監視プロセスがなく装置の操作が不能になるという状況をなくすることが可能である。

【００２９】以下、本発明の実施の形態を図面を参照して具体的に説明する。図１は、本発明のエラー制御デバイスを使用したＦＡＸ情報サービス装置のソフトウエア構成を示している。１０１はＦＡＸ情報サービス装置であり、このＦＡＸ情報サービス装置１０１のメモリ空間は、後に詳述するように、複数のプロセスが実行されるユーザ空間１０２と、基本オペレーティングシステム（以下、ＯＳという）が実行される基本ＯＳ空間１０３と、から構成されている。更に、この基本ＯＳ空間１０３では、エラー制御デバイスドライバ１０４とそれ以外のデバイスドライバ１０５、例えば、複数チャネルファクシミリ通信ボードドライバ等との双方が動作し、ユーザ空間１０２では、監視プロセス１０６とポート制御プロセス（１０７～１０９）との双方が動作するようになっている。

【００３０】さて、ＦＡＸ情報サービス装置１０１は、電話回線と１対１に対応するＮ個のポート制御プロセス（１０７から１０９）を有し、複数のファクシミリ文書を複数の情報ボックス（図示せず）に分類して蓄積している。ポート制御プロセス（１０７から１０９）はユーザから来る呼の着信を待ち、着呼後にユーザが所定の情報ボックスの情報の取出しの指示をした場合には、ＰＢトーン等で受信し、指定された情報ボックスに蓄積されたファクシミリ文書を送信する制御を行う。ユーザが情報ボックスへの登録を指示した場合には、ポト制御プロセスはファクシミリ画像を受信し、指定の情報ボックスに蓄積する制御を行う。

【００３１】このポート制御プロセス（１０７～１０９）が動作中にエラー発生を検出した場合には、各々のポート制御プロセス（１０７～１０９）に対応して設けられた被監視デバイスファイル（１１０～１１２）にアクセスしてエラー通知を行なう。一方、監視プロセス１０６はＦＡＸ情報サービス装置の起動、停止処理を行う他に、監視デバイスファイル１１０にアクセスしてポート制御プロセスの実行監視を行うようになっている。

【００３２】エラー制御デバイスドライバ１０４は、基本ＯＳ空間１０３に、作業領域として監視デバイスファイル１１０に対応する監視デバイスワークメモリ１１４と、被監視デバイスファイル（１１０～１１２）に対応する被監視デバイスワークメモリ（１１５～１１７）を有し、これらのワークメモリは、それぞれ対応するデバイスファイルの状態を記述する領域を有している。ＦＡＸ情報サービス装置１０１が起動するときに、基本ＯＳはエラー制御デバイス１０４の初期化処理を呼び出し、前記初期化処理はワークメモリ（１１４～１１７）を割り当て、各ワークメモリの状態フィールドをクローズに設定する。

【００３３】また、エラー処理デバイスドライバ１０４

は、使用開始手段１１８と、監視手段１１９と、監視停止手段１２０と、登録手段１２１と使用終了手段１２２とを有しており、使用開始手段１１８は対応するデバイスファイルがオープンされたときに基本ＯＳにより呼び出され、使用終了手段１２２は対応するデバイスファイルがクローズされたときに基本ＯＳにより呼び出されるプログラムであり、また、監視手段１１９と監視停止手段１２０と登録手段１２１とは、基本ＯＳが提供するシステムコール（以下では入出力制御命令と呼ぶ）により実行される固有のプログラムである。

【００３４】以上のようなソフトウエア構成を採るＦＡＸ情報サービス装置におけるエラー制御デバイスドライバによるエラー処理フローを図２乃至図８を用いて説明する。

【００３５】最初に使用開始手段１１８について説明する。使用開始手段１１８は、エラー制御デバイスドライバの一部を構成するプログラムであり、システムが立ち上がり、いずれかのプロセスが開始されてデバイスファイルがオープンされた時に基本ＯＳにより呼出され、そのプロセスに対応するデバイスファイルのワークメモリ（１１４～１１７）の状態フィールドをオープン状態にするものである。図２は、その動作の詳細フローを示している。

【００３６】監視プロセス１０６またはポート制御プロセス（１０７～１０９）が、それぞれ、監視デバイスファイル１１３または被監視デバイスファイル（１１０～１１２）をオープンしたときに、基本ＯＳはオープンされたそのデバイスファイルをパラメータとして、エラー制御デバイスドライバ１０４の使用開始手段１１８を呼び出す。

【００３７】使用開始手段１１８は、パラメータで指定されたデバイスファイルが監視デバイスファイル１１３かどうかをチェックし（ステップ２０１）、監視デバイスファイル１１３であれば、既に監視デバイスファイル１１３がオープンされているか否かを監視デバイスワークメモリ１１４の状態フィールドを参照することによりチェックする。その状態フィールドがオープンの状態であれば、既に他のプロセスが監視デバイスファイルをオープンしているということなので、ステップ２０３でエラーをリターンする。逆に、その状態フィールドの値がクローズであれば、ステップ２０４で状態フィールドをオープンに設定し、ステップ２０５にて成功をリターンする。

【００３８】パラメータ指定のデバイスファイルが監視デバイスファイル１１３でなければ、被監視デバイスファイル（１１０～１１２）であるので、ステップ２０６において前記ファイルのデバイス番号をチェックする。デバイス番号が不正であればエラーをリターンするが（ステップ２０８）、デバイス番号が正しければ、そのデバイスファイルに対応するワークメモリ（１１５～１１７）の状態フィールドをチェックする（ステップ２０９）。その状態フィールド値がオープンであればエラーをリターンし（ステップ２０８）、クローズであればその状態フィールドをオープンに更新して（ステップ２１０）、成功をリターンする（ステップ２１１）。

【００３９】以上一連の処理が使用開始手段１１８の動作であり、これにより、監視デバイスファイル１１３及び実行される被監視デバイスファイル（１１０～１１２）のワークメモリ（１１４～１１７）がオープン状態となり、エラー処理の監視プロセスの実行準備が整う。

【００４０】次に、監視手段１１９及び監視停止手段１２０について説明する。監視手段１１９及び監視停止手段１２０は、エラー制御デバイスドライバ１０４の一部を構成するプログラムであり、システム起動時に、監視プロセス１０６により基本ＯＳを介して呼出され、その間被監視デバイスファイル（１１０～１１２）のエラー監視を実行するものである。

【００４１】図３は、監視手段１１９の動作の詳細フローを示している。監視プロセス１０６がオープンした監視デバイスファイル１１３に対して入出力制御コールを実行して監視手段１１９を呼び出すと、基本ＯＳは、監視デバイスファイル１１３とその監視デバイスファイル１１３が指定した受信バッファアドレスをパラメータとして監視手段１１９を実行する。

【００４２】まず、監視手段１１９は、パラメータで指定されたデバイスファイルが監視デバイスファイル１１３かどうかをチェックし（ステップ３０１）、監視デバイスファイル１１３でなければエラーをリターンする（ステップ３０２）。監視デバイスファイル１１３であれば、監視デバイスワークメモリ１１４の状態フィールドを監視中に設定するとともに（ステップ３０３）、受信バッファアドレスを指定アドレスに設定して受信待ち状態に入りエラー検出を待つ（ステップ３０４）。

【００４３】この受信バッファに何等かの入力があれば待ち状態を終了し実行状態に入り、監視デバイスワークメモリ１１４の状態フィールドを再びオープン状態に設定した後（ステップ３０５）、受信バッファの内容を参照し監視停止手段が実行されたかどうかを判断する（ステップ３０６）。受信バッファに後述の監視停止手段１２０からの監視停止命令が設定されている場合は、監視停止をリターンし（ステップ３０７）、受信バッファに監視停止命令ではなくエラー情報が設定されていれば成功をリターンする（ステップ３０８）。このエラー検出後の処理は監視プロセス１０６が行なう。

【００４４】図４は監視停止手段１２２の動作の詳細フローを示している。監視プロセス１０６がオープンした監視デバイスファイル１１３に対して、監視停止手段１２０を指定した入出力制御コールを実行すると、基本ＯＳは監視デバイスファイル１１３をパラメータとして監視停止手段１２０を実行する。監視停止手段１２０は、

パラメータで指定されたデバイスファイルが監視デバイスファイル１１３であるか否かをチェックし（ステップ４０１）、監視デバイスファイル１１３でなければエラー処理とし（ステップ４０２）、監視デバイスファイル１１３であれば監視デバイスワークメモリ１１４の状態フィールド値が監視中であるかチェックする（ステップ４０３）。状態フィールド値が監視中でなければエラー処理とし（ステップ４０２）、監視中であれば監視デバイスワークメモリ１１４のバッファアドレスフィールドで指定された受信バッファに監視停止をライトし（ステップ４０４）、基本ＯＳインターフェースであるＷＡＫＥＵＰ処理をコールして（ステップ４０５）、受信待ちの状態になっている監視手段１１９を待ち状態から監視停止状態に変更し、成功をリターンする（ステップ３０７）。

【００４５】以上一連の処理が監視手段１１９及び監視停止手段１２０の動作であり、エラー監視が必要な処理に対して監視手段１１９の起動をかけるだけで、エラーが検出されるか監視停止処理が実行されるまでエラー監視が続行されることとなる。また、監視停止処理も監視手段１１９とは独立のプログラムである監視停止手段１２０により実行される。

【００４６】次に、登録手段１２１について説明する。登録手段１２１は、エラー制御デバイスドライバ１０４の一部を構成し、エラー監視の対象となるポート制御プロセス（１０７乃至１０９）の起動時に、各プロセスを特定するための情報を基本ＯＳ空間上のワークメモリに登録しておくためのプログラムであり、図５は、その動作の詳細フローを示している。

【００４７】ポート制御プロセス（１０７乃至１０９）がオープンされた被制御デバイスファイル（１１０～１１２）に対して入出力制御コールを実行し、登録手段を呼び出すと、基本ＯＳは、被制御デバイスファイル（１１０～１１２）とポート制御プロセス（１０７乃至１０９）が提供するプロセスＩＤとポート番号とを含む情報を受信する受信バッファのアドレスをパラメータとして登録手段１２１を呼び出す。登録手段１２１は、パラメータのデバイスファイルが被監視デバイスファイル（１１０～１１２）か否かをチェックし（ステップ５０１）、被監視デバイスファイル（１１０～１１２）でない場合はエラー処理をし（ステップ５０２）、被監視デバイスファイル（１１０～１１２）である場合は、そのデバイスファイルの番号をチェックする（ステップ５０３）。デバイスファイル番号が不正であればエラー処理をし（ステップ５０２）、デバイスファイル番号が正しければ、前記受信バッファの内容を対応する被監視デバイスワークメモリ（１１５乃至１１７のメモリのうち１つ）にコピーして成功をリターンする（ステップ５０４、テスップ５０５）。

【００４８】以上一連の処理が登録手段１２１の動作であり、エラーが発生するプロセスを特定する情報を基本ＯＳのワークエリア上に予め記述しておくことにより、監視プロセスがエラー発生時にその情報を受け取り、エラー個所を認識できることになる。

【００４９】次に、使用終了手段１２２について説明する。使用終了手段１２１は、エラー制御デバイスドライバ１０４の一部を構成するプログラムであり、いずれかのプロセスが終了しデバイスファイルがクローズした時に基本ＯＳにより呼出され、そのプロセスに対応するデバイスファイルのワークメモリ（１１４～１１７）の状態フィールドをクローズ状態にして、エラー発生プロセスを特定する情報を監視手段１１９の受信バッファにコピーすることにより、スリープ状態にあった監視手段１１９を実行状態にするものである。既に説明したように、この監視手段１１９によりエラー情報が監視プロセス１０６に通知されることでエラー検出が行なわれる。図６はその動作の詳細フローを示している。

【００５０】監視プロセス１０６またはポート制御プロセス（１０７～１０９）が、それぞれ、オープン中の監視デバイスファイル１１０または被監視デバイスファイル（１１０～１１２）をクローズしたときに、基本ＯＳはそのデバイスファイルをパラメータとして、エラー制御デバイスドライバ１０４の使用終了手段１２２を呼び出す。使用終了手段１２２は、クローズされたデバイスファイルが被監視デバイスファイル１１３かどうかをチェックする（ステップ６０１）。

【００５１】クローズされたデバイスファイルが、被監視デバイスファイル（１１０～１１２）でない（監視デバイスファイルである）場合は、監視デバイスワークメモリ１１４の状態フィールドが監視中であるか否かをチェックし（ステップ６１０）、監視中でなければ成功をリターンする（ステップ６１１）一方、監視中であれば基本ＯＳのリブートインターフェースによりＦＡＸ情報サービス装置１０１をリブートする（ステップ６１２）。

【００５２】このシステムリブート（ステップ６１２）が実行されるのは、監視中の監視プロセス１０６が何らかの障害により異常終了し、その際基本ＯＳによってオープン状態であった監視デバイスファイル１１３がクローズされる場合である。このように、監視中の監視プロセス１０６が異常終了したときに、エラー制御デバイスドライバ１０４が装置をリセットする機能を設け、装置リブートの際に監視プロセスを自動的に立ち上げるように設定することにより、監視プロセスの存在を定期的にウオッチする処理と、監視プロセスが異常終了した際のエラー処理を別途設けることなく、システムが制御不能な状態で放置されるという状況を防止することが可能となる。

【００５３】次に、クローズされたデバイスファイルが被監視デバイスファイル（１１０～１１２）である場合

について説明する。使用終了手段１２２は、デバイス番号のチェックを行い（ステップ６０２）、デバイス番号が不正であればエラー処理を行ない（ステップ６０３）、デバイス番号が正しければクローズされたデバイスファイルに対応する被監視デバイスワークメモリ（１１５から１１７のうちの１つ）の状態フィールドをクローズに設定し（ステップ６０４）、監視デバイスワークメモリ１１４の状態フィールドが監視中か否かをチェックする（ステップ６０５）。監視デバイスワークメモリ１１４の状態フィールドが、監視中でない場合は、成功をリターンし（ステップ６０８）、監視中である場合は、監視デバイスワークメモリ１１４の受信バッファアドレスフィールドで指定されている受信バッファに、クローズされた被監視デバイスファイル（１１０～１１２）に対応する被監視デバイスワークメモリ（１１５から１１７のうちの１つ）に格納されている情報（例えば、プロセスＩＤとポート番号）をコピーした後（ステップ６０６）、基本ＯＳインターフェースであるＷＡＫＥＵＰをコールし（ステップ６０７）、受信待ちの状態になっている監視手段１１９を待ち状態から実行状態に変更し、成功をリターンする（ステップ６０８）。

【００５４】以上一連の処理が使用終了手段１２２の動作であるが、このように、被監視デバイスファイル（１１０～１１２）がクローズされるのは、ポート制御プロセス（１０７～１０９）がエラーを検知して前記デバイスファイルをクローズする場合と、当該プロセスが例外等で異常終了し、基本ＯＳがオープン状態にある前記デバイスファイルをクローズする場合と、がある。前者の場合には通常のエラー処理が実行されるが、後者のように、監視対象のポート制御プロセス（１０７～１０９）が異常終了した場合であっても、既に説明した登録手段により被監視デバイスワークメモリ（１１５～１１７）に予め格納してある情報が監視デバイスファイル１１３に必ず登録されるため、障害情報を確実に監視プロセスに通知することが可能となる。

【００５５】従って、監視プロセスが定期的に被監視プロセスの状態をチェックする必要がなくなり、エラー処理部のプログラム量を飛躍的に減らすことが可能になる。また、エラー検知にプロセス間通信を使用しないので装置の計算資源の浪費を避けることができる。

【００５６】以上のようなソフトウエア構成をもつエラー制御デバイスドライバを使用して、監視プロセス１０６がポート制御プロセス（１０７～１０９）の状態監視、エラー処理を実行するフローを、図７を用いて具体的に説明する。この監視プロセス１０６は、ＦＡＸ情報サービス装置１０１の起動、停止、ポート制御プロセス（１０７～１０９）の状態監視を行い、ポート制御プロセスが異常終了した場合、エラー処理として前記プロセスの再起動等を実行するようになっている。

【００５７】まず、システムの起動処理が実行されると（ステップ７０１）、ポート制御プロセス（１０７～１０９）を必要数（電話回線数）起動するとともに、監視デバイスファイル１１３をオープンして（ステップ７０２）、エラー監視用スレッドを生成する（ステップ７０３）。生成されたメインスレッドでは、ステップ７１２以下の監視プロセス１０６本来の処理、つまり、システム停止イベントの処理を実行する。イベント受信待ちの状態で（ステップ７１２）、イベントを受信すると、その受信イベントがシステム停止か否かをチェックし（ステップ７１３）、それがシステム停止イベントでない場合にはイベントの処理（通常の処理）を実行して（ステップ７１４）、再びイベント受信待ちの状態に復帰する（ステップ７１２）。受信イベントがシステム停止イベントである場合には、、監視停止手段を指定した入出力制御コールを発行して監視停止手段により監視処理を停止し（ステップ７１５）、システム停止処理を実行し（ステップ７１６）、監視プロセスを停止する（ステップ７１７）。

【００５８】一方、生成されたサブスレッドでは以下の処理が実行される。エラーが発生した場合の障害情報を受け取るバッファアドレスをパラメータとして、監視手段を指定した入出力制御コールを発行して監視手段１１９を呼出し（ステップ７０５）、エラーが発生するのを待つ。ポート制御プロセス（１０７～１０９）が異常終了した場合と監視プロセス１０６が監視停止手段１２０を呼び出した場合には、監視手段１１９は待ち状態から実行状態に遷移し、ステップ７０５から処理が再開される。

【００５９】次いで、監視手段１１９は重大なエラーが発生したのかを、ファクシミリ通信ボードのハードウエア等をアクセスしてチェックする（ステップ７０６）。重大エラーが発生した場合には装置の運用を停止するために、システム停止イベントをメインスレッドに送信し（ステップ７０７）、エラー監視スレッドを終了する（ステップ７０８）。

【００６０】ステップ７０６でのチェックの結果、受信したエラーが重大エラーでないならば、監視停止手段が実行されたのかをチェックし（ステップ７０９）、監視停止が実行されたのであればスレッドを終了する（ステップ７１０）。また、監視停止が実行されたのではなく、受信したエラーが復旧可能なエラーである場合は、復旧処理を行った後（ステップ７１１）、ステップ７０５にジャンプしてエラー監視を続行する。復旧処理は、例えば、障害により制御プロセスが停止したポート番号を受信バッファから参照し、そのポートのポート制御プロセスを再起動することにより行なう。

【００６１】以上一連の処理が監視プロセス１０６の動作であるが、このように監視プロセス１０６全体をメインスレッドとサブスレッドに分割し、通常のイベント処理とシステム停止処理とをメインスレッドで実行する一方、監視手段１１９を呼出してエラー検出をしエラーリカバリを実行する処理をサブスレッドで実行することにより、監視プロセス１０６がエラー監視をする処理としない処理とを明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を設ける必要がなくなる。更に、サブスレッドで監視手段１１９を呼び出す（ステップ７０５）のみでエラー監視が可能となり、プロセス間通信等を使用して定期的に被監視プロセスの異常終了を検知をする必要がなくなるため、エラー検知処理部が非常に簡素化される。

【００６２】次に被監視プロセスであるポート制御プロセス（１０７～１０９）の処理フローについて説明する。図８は、ポート制御プロセス（１０７～１０９）のエラー通知フローを示している。まず、制御する通信ポートの初期化処理を行い（ステップ８０１）、制御するポート番号を獲得して被監視デバイスファイル（１１０～１１２）をオープンする（ステップ８０２）。次いで、登録すべき情報の一つであるプロセスＩＤを基本ＯＳから獲得し（ステップ８０３）、入出力制御コールを基本ＯＳに発行して登録手段を呼び出す（ステップ８０４）。更に、本来の処理であるファクシミリ通信処理を実行した後（ステップ８０５）、エラーチェックを行い（ステップ８０６）、エラーがない場合には、通常処理である通信処理（ステップ８０５）を繰り返す。

【００６３】エラーを検知した場合には、被監視デバイスファイル（１１０～１１２）をクローズし（ステップ８０７）、ポート制御プロセス（１０７～１０９）を終了する（ステップ８０８）。このクローズ処理（ステップ８０７）により図６の使用終了手段１２２が呼び出され、ステップ８０４で登録手段１２１に登録しておいた情報が監視プロセスのバッファにコピーされ（ステップ６０６）、監視手段を実行状態に設定する（ステップ６０７）、ことによりエラー情報が監視プロセス１０６に通知される。

【００６４】以上一連の処理がポート制御プロセス（１０７～１０９）の動作であり、クローズ処理により障害情報を監視プロセスに通知するようにしたことにより、例外等が発生し被監視プロセスが消滅した場合においても、基本ＯＳがオープンされている被監視デバイスファイルをクローズするので、使用終了手段が確実にコールされ、監視プロセスに障害情報が確実に通知される。従って監視プロセスが被監視プロセスの実行状況を定期的にプロセス間通信等により監視する必要がなくなり、エラー処理部が非常に簡単となり、さらに、計算資源の浪費を押さえることができることとなる。

【００６５】

【発明の効果】以上の説明から明らかなように、本発明により、監視プロセスが定期的に被監視プロセスの状態をチェックすることなく、確実簡易にエラー通知を行うことが可能となるため、エラー処理部のプログラム量を飛躍的に減らすことが可能になり、また、エラー検知にプロセス間通信を使用しないので装置の計算資源の浪費を避けることができる。

【００６６】更に、具体的には、エラー検出に必要な情報を被監視プロセスとは異なる場所に保持しておき事後的に監視プロセスに通知することにより被監視プロセスが異常終了した場合でも確実にエラー検出を行ない得る。　また、システム起動時に際しては確実にエラーチェックの準備が整うことになり、被監視プロセス起動直後にそのプロセスのエラー監視が可能になる。

【００６７】また、エラー通知に必要な情報を被監視プロセス終了時に確実に監視プロセスに通知することが可能となる。

【００６８】また、監視手段がエラー検出を開始、続行するため、監視手段を呼出すだけで全ての監視対象プロセスで生じたエラーを検知することが可能となる。

【００６９】また、監視手段によりエラー監視を開始し、監視終了手段によりエラー監視を中断でき、監視不要な処理に対してのエラー監視、エラー通知を避けることが可能となる。

【００７０】また、エラー監視をする処理としない処理を明確に区別することが可能になるため、エラー監視をしない処理において、不必要なエラー処理を実行する必要がなくなる。

【００７１】また、被監視プロセスが異常終了した場合でも確実にエラー検出,エラー通知を行ない得る。

【００７２】また、被監視プロセスの状態監視用のデバイスファイルの利用によりエラー検知を確実に行ない得る。

【００７３】更に、リセットにより必要となるプロセスを全て再生できるので、監視プロセスがなく装置の操作が不能になるという状況をなくすることが可能となる。

【図面の簡単な説明】

【図１】本発明のエラー制御デバイスドライバを使用したＦＡＸ情報サービス装置ソフトウエア構成図

【図２】本発明のエラー制御デバイスドライバの使用開始手段詳細フロー図

【図３】本発明のエラー制御デバイスドライバの監視手段詳細フロー図

【図４】本発明のエラー制御デバイスドライバの監視停止手段詳細フロー図

【図５】本発明のエラー制御デバイスドライバの登録手段詳細フロー図

【図６】本発明のエラー制御デバイスドライバの使用終了手段詳細フロー図

【図７】本発明の監視プロセスの処理フロー図

【図８】本発明のポート制御プロセスの処理フロー図

【図９】従来例によるエラー監視プロセスの詳細フロー図

【図１０】従来例による被エラー監視プロセスの詳細フ

ロー図

【符号の説明】

104　エラー制御デバイスドライバ

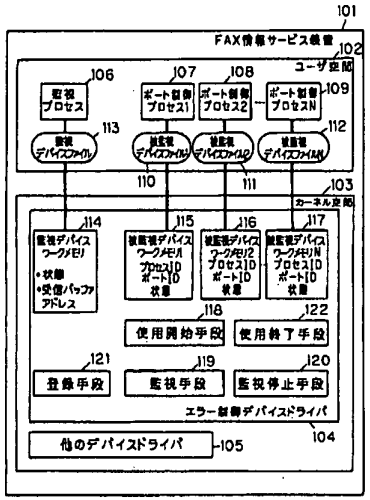
106　監視プロセス

107　ポート制御プロセス

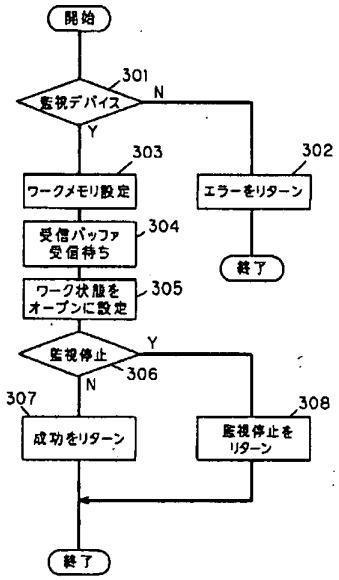
108　ポート制御プロセス

109　ポート制御プロセス

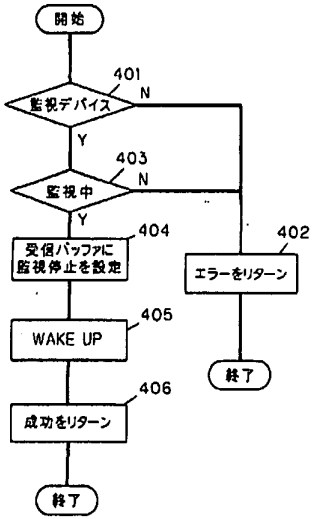
【図1】



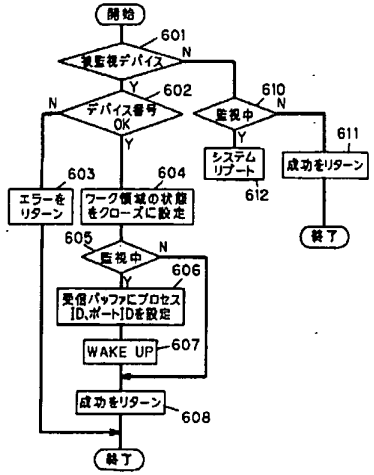
【図3】



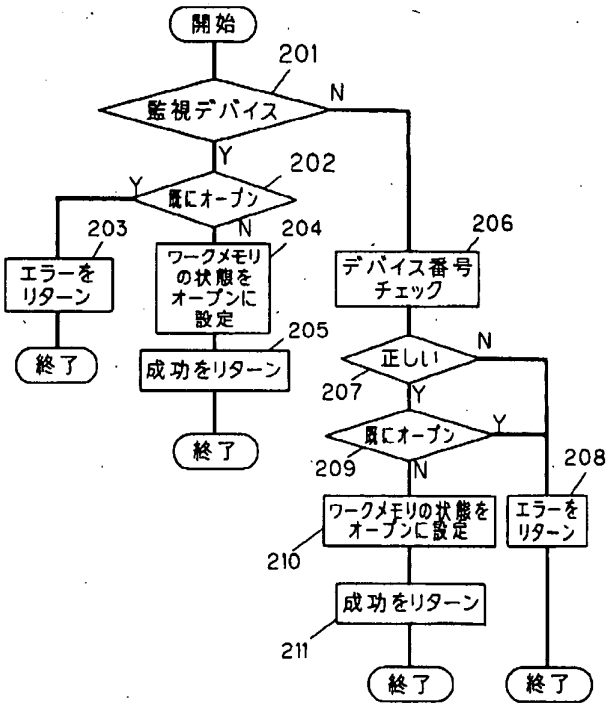
【図4】



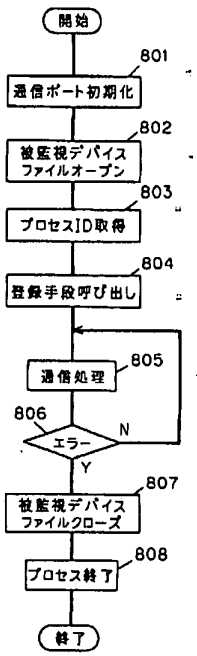
【図6】



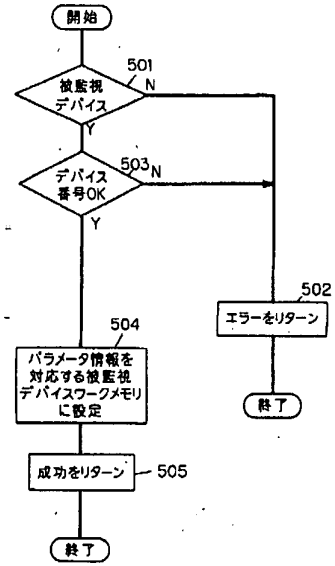
【図2】



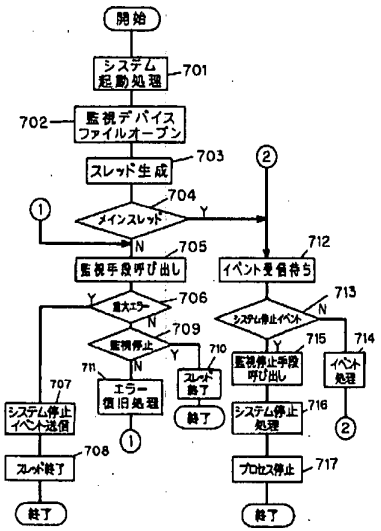
【図8】



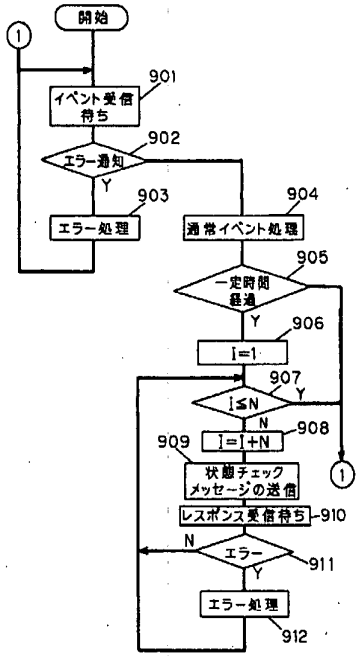
【図5】


開始
501
被監視デバイス
N
Y
503
デバイス番号OK
N
Y
502
エラーをリターン
504
パラメータ情報を対応する被監視デバイスワークメモリに設定
505
成功をリターン
終了
終了

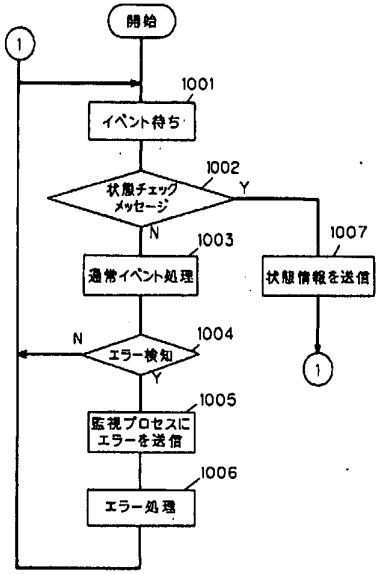
【図7】


開始
701
システム起動処理
702
監視デバイスファイルオープン
703
スレッド生成
704
メインスレッド
Y
N
705
監視手段呼び出し
706
重大エラー
Y
N
707
システム停止イベント送信
708
スレッド終了
終了
709
監視停止
Y
N
710
スレッド終了
711
エラー復旧処理
終了
712
イベント受信待ち
713
システム停止イベント
N
Y
714
イベント処理
715
監視停止手段呼び出し
716
システム停止処理
717
プロセス停止
終了
①
②

【図9】


開始
901
イベント受信待ち
902
エラー通知
Y
903
エラー処理
904
通常イベント処理
905
一定時間経過
Y
906
I=1
907
I≦N
Y
N
908
I=I+N
909
状態チェックメッセージの送信
910
レスポンス受信待ち
911
エラー
N
Y
912
エラー処理
①

【図10】


開始
①
1001
イベント待ち
1002
状態チェックメッセージ
N
Y
1003
通常イベント処理
1007
状態情報を送信
1004
エラー検知
N
Y
①
1005
監視プロセスにエラーを送信
1006
エラー処理